# MINMAX

## A General Purpose Adaptive Iterator for Nonlinear Problems

J. Campbell *
W. E. Moore**
H. Wolf**

Nas 9-2527

June 30, 1964

*Manned Spacecraft Center, N. A. S. A.
**Analytical Mechanics Associates, Inc.

Analytical Mechanics Associates, Inc.
Uniondale, L. I., N. Y.

ACKNOWLEDGEMENT

The iterator described in the following is included in the Mission Analysis Program developed by Analytical Mechanics Associates, Inc. in cooperation with the Mission Analysis Branch of the Manned Spacecraft Center under the general direction of M. V. Jenkins. It is in the nature of such a program that it embodies ideas contributed by many people and individual acknowledgements are not possible.

## 1. Introduction

One of the central problems in mission analysis, in real time as well as in preliminary design, is the construction of a trajectory which permits a set of mission objectives to be accomplished subject to numerous operational and other constraints. Mission objectives as well as constraints are usually expressed as functions of the position and velocity vectors, called the "state vector" for short, at various times. These times may either be given directly or be defined implicitly as the time when a given geometric or dynamic configuration is achieved.

The state vector at any time may be computed as a function of a set of initial conditions and a number of control parameters such as duration, magnitude and direction of thrusts. Thus, a split boundary value problem results. In general, this functional relationship is complicated and highly nonlinear, and in many cases no closed form expression of one set in terms of the other is available. The difficulty inherent in the solution of such split boundary value problems is compounded by the fact that the number of constraints and objectives may be either larger or smaller than the number of independent parameters to be determined. Further, it is often desirable to select an optimum trajectory (defined according to some criterion, such as maximum payload, shortest mission time, etc.) in cases where many trajectories are possible. If such a program is to be used in real time, it

must further be capable of producing a solution independent of the availability of a good first guess. The program thus must be capable to start with any "neutral" first guess and proceed to a solution.

## 2. Summary

A general formulation has been developed which contains all the problems described in paragraph 1, and a method of solution was devised to cope with this general formulation. This method has been applied to the solution of a great variety of problems and many cases and has been proven to have great power and flexibility. Essentially the formulation consists in reducing all the problems described to the problem of finding the minimum length of a vector. The construction of this residual vector is detailed in Section 4.

The method of solution is an iteration technique which successively improves the trajectory. These small corrections are determined by a local linear approximation. Since, in fact, the problem is highly nonlinear, it is necessary to restrict the size of the corrections to the linear range. This range limitation further must be used to modify the direction of the correction vector, for this is the only way to guarantee continued improvement of the solution from iteration to iteration.

An inhibiting parameter is used to effect the limitation of correction size. Control of the inhibitor to achieve these objectives is described in Section 5. Essentially, the method used is a modification and extension of a method described by Morrison (Ref. 1) and Marquardt (Ref. 2).

Since the "residual vector" which is being minimized consists of components of dissimilar character, scale factors must be introduced to properly define the length of such a vector. These scale factors can then be adjusted to control the rate of convergence to a solution and give varying emphasis to the constraints according to their severity. They may also be used to distinguish between two phases of the iteration, the first being the "selection phase" where an acceptable trajectory is obtained, the second an "optimization phase" where the best of many acceptable trajectories is found. The handling of the scale factors in the various phases is described in Sections 6, 7, and 8.

## 3. Notation

In what follows, lower-case letters will refer to (column) vectors of various dimensions. If a vector has n(m) elements it will be referred to as an n-vector (m-vector). Upper-case letters are matrices. A prime (') applied to either a matrix or a vector denotes transposition. Sometimes capital letters will represent scalar values. The usage of each letter will be made clear as it is introduced.

## 4. Formulation

For the sake of convenience, we lump both initial values and control parameters together under the heading of independent variables and we represent the end conditions by dependent variables. For the problem to be solved, let

$$y = f(x) \tag{1}$$

stand for the relationship between the independent variables constituting the m-vector $x$ and the dependent variables constituting the n-vector $y$. It is assumed that a procedure exists to compute $y$ if $x$ is known. Let the vector of desired values of the constraint variables be denoted by $\bar{y}$. Then the problem is to find a vector of independent values $\bar{x}$ such that

$$\bar{y} = f(\bar{x}) \tag{1a}$$

$y$ and $f$ are n-vectors and $x$ is an m-vector. Note that, as far as this technique is concerned, there is no restriction on the relative sizes of $m$ and $n$. However, in setting up any given problem, care must be taken that the constraints do not permanently overdetermine the system and that the constraints are at least plentiful enough to permit an interpretation of any result (see Section 6) in case the system is underdetermined.

The symbol $f(x)$ stands for any method of generating an end condition if independent variables are known. The $f$'s may take many forms. They

may be closed form solutions to differential equations; they may be differential equations to be solved by numerical integration; or sometimes they may be relatively simple algebraic relationships. In any case, it is expected that most of the f's represent highly nonlinear relationships between x and y.

This nonlinearity prevents a direct solution of the problem. The technique starts by replacing the nonlinear problem with a linear one, by an expansion about a nominal trajectory. On the one hand, we get the advantage of having a direct solution; on the other hand, we have the disadvantage that the solution so obtained is not really the solution to the original problem. To allow for this, an iteration technique is applied to the solution of the linear problem.

Thus, consider the matrix of partial derivatives of the form $\frac{\partial f}{\partial x}$. Let this matrix be denoted by P. We replace (1) by its first-order Taylor expansion at x.

$$\bar{y} = f(\bar{x}) = f(x) + P(\bar{x} - x) \quad ,$$

or (2)

$$\bar{y} - y = P(\bar{x} - x) \quad .$$

In principle if $n = m$, that is, if the number of dependent variables equals the number of independent variables, and if P is nonsingular, so that $P^{-1}$ exists, we can write

$$\Delta x = P^{-1} (\bar{y} - y) \tag{3}$$

and

$$\bar{x} = x + \Delta x \tag{4}$$

Equation (2) represents a linear approximation to equation (1). Consequently when the new value of $x$ given by equation (4) is used to compute $y$ by equation (1), the value of $y$ cannot be expected to be the desired value. Indeed, it is not even guaranteed that the new length of the $\Delta y$ vector is less than the previous length. All of this is a reflection of the fact that, for too large a vector $\Delta x$, the linear estimate may not be adequate and the elements of the matrix of partial derivatives have values differing from their old values due to second order effects. In some lucky cases this is not too serious, and by iterating the computations represented by (1), (3), and (4), a solution of (1a) is obtained. For additional reliability, however, other measures are adopted. At the same time, we desire a method for handling cases where $m$, the number of $x$-parameters, is not equal to $n$, the number of $y$-parameters. Both of these objectives can be achieved by restating the problem as follows.

Consider the scalar

$$R = (\Delta y)' W_y \Delta y \tag{5}$$

where $W_y$ is an $n \times n$ diagonal matrix of positive scale factors making the components of the vector $\Delta y$ compatible among themselves. The

number R is, of course, the weighted sum of the squares of the residuals in the desired quantities. Because the scale factors are positive, R can never be negative. On the other hand, if all of the residuals can be made zero, R can be zero. Thus, in either case, we desire to reduce R to a minimum. Equation (2) says that, for any vector $\Delta x$, the predicted new $\Delta y$ will be:

$$\Delta y = \bar{y} - y - P \Delta x \tag{6}$$

Formula (3) may produce a $\Delta x$ vector which is quite large, especially when $\bar{y} - y$ is large or P is ill-conditioned. Due to nonlinear effects, such a $\Delta x$ vector creates quite wild effects of $\Delta y$. To prevent this, we impose the side condition that

$$S = (\Delta x)' W_x \Delta x \leq S_o \quad . \tag{7}$$

where $W_x$ is an $m \times m$ diagonal matrix of scale factors making the components of the $\Delta x$ vector compatible and $S_o$ is a constant chosen to guarantee the approximate validity of the linear estimate. For any vector x we construct $\Delta y$ as in (6) and compute R from it by equation (5). We attempt to choose $\Delta x$ so as to minimize R within the constraint (7). This results in the formula for $\Delta x$ given by equation (8):

$$\Delta x = (P' W_y P + \lambda W_x)^{-1} P' W_y \Delta y \tag{8}$$

where $\lambda$ is a parameter chosen to enforce (7). It is called the inhibitor.

Morrison has shown that as $\lambda$ increases, the length $S$ decreases. Thus, it is always possible to choose $\lambda$ so big that $S$ satisfies the inequality (7). It is, however, not necessary to determine $\lambda$ so as to obtain an equality in equation (7) since too large a value of $\lambda$ and, consequently, too small a value of $S$ will at most cost a few additional iterations (see also Section 5). See the next section for the mechanism of this choice.

Apart from the a priori limitation of the permissible size of $\Delta x$ expressed by equation (7), the inhibitor $\lambda$ is also used to achieve an actual reduction of the residual vector $R$ [equation (5)] in cases where the limitation expressed by equation (7) is not adequate to guarantee linear behavior.

Now the $\Delta x$ obtained from (8) is used in (4) to get a new value of $x$ so that the process can be repeated. Thus, the iteration procedure consists of repetitively computing the following items:

(1) $y$ from equation (1)

(2) the $P$ matrix

(3) the value of the inhibitor $\lambda$ (see Section 4)

(4) the value of the vector $\Delta x$ from (8)

(5) the new value of $x$ from (4)

Since the derivation of (8) is a least-squares procedure anyway, it can be applied to any problem for which $n \geq m$. In addition, if $n < m$, and if one of the dependent variables is to be optimized, (8) can be used to find $x$ so as to attain all of the other dependent variables and come as close to the optimized one as possible. Thus, there may be more constraints, fewer constraints, or the same number of constraints as there are unknowns in the problem.

5. Mechanization of the Inhibitor Control

The theory leading to equation (8) states that the unknown parameter $\lambda$ should be determined to satisfy the inequality

$$S[\Delta x(\lambda)] = S(\lambda) = [\Delta x(\lambda)]' W_x \Delta x(\lambda) \leq S_o \tag{7}$$

As stated above, $S$ is a decreasing function of $\lambda$; indeed $S(\lambda) \to 0$ as $\lambda \to \infty$. Generally, it would be most efficient to take a step in $\Delta x$ as big as (7) will allow, or, in other words, to make $\lambda$ as small as (7) will allow.

Inequality (7), however, does not lend itself to easy solution and such a solution is not necessary. Hence, a strategy for choosing $\lambda$ at each iteration step has to be developed. A good global strategy must be based on two conditions:

($\alpha$) At each stage

$$S(\lambda) \leq S_o \tag{7}$$

($\beta$) If $R^{(p)}$ denotes the value of $r$ at the $p^{th}$ iteration

$$R(p) \leq R^{(p-1)} \tag{9}$$

This second condition is a statement of the obvious fact that minimizing $R$ requires reducing $R$. Most of the problems to which the iteration has been applied have been overdetermined, in the sense that $n \geq m$. In case

this happens, where $R$ approaches its minimum value it begins to move very slowly in response to $\Delta x$, so that care must be taken not to overshoot the point where the smallest value is attained. Condition (9) assures this.

A strategy which satisfies conditions $(\alpha)$ and $(\beta)$ above and efficiently controls the necessary number of iterations is the following:

Let

$$a > 1 \quad \text{and} \quad b \geq a \tag{10}$$

Let $\lambda^{(p-1)}$ be the value of $\lambda$ from the previous iteration. For any value of $\lambda$, equation (8) furnishes $\Delta x(\lambda)$. Equation (7) defines $S(\lambda)$. Equation (1) with $x + \Delta x(\lambda)$ for $x$ defines $y(\lambda)$:

$$y(\lambda) = f[x + \Delta x(\lambda)]$$

Thus

$$R(\lambda) = [\bar{y} - y(\lambda)]' W_y [\bar{y} - y(\lambda)]$$

is the value of $R$ corresponding to $\lambda$.

(a) Set $\lambda = \dfrac{\lambda^{(p-1)}}{b}$.

(b) Compute $S(\lambda)$ and compare it with $S_o$.

(c) If $S(\lambda) \leq S_o$, proceed to (d).

If $S(\lambda) > S_o$, replace $\lambda$ by $a\lambda$ and repeat from step (b). The final value $\lambda$ is $a^q\lambda$ where $q$ is the smallest integer such that $S(a^q\lambda) \leq S_o$.

(d) Compute $R(\lambda)$ and compare it with $R[\lambda^{(p-1)}]$.

(e) If $R(\lambda) \leq R[\lambda^{(p-1)}]$, set $\lambda^{(p)} = \lambda$.

If $R(\lambda) \geq R[\lambda^{(p-1)}]$, replace $\lambda$ by $a\lambda$ and repeat from step (d). The final value $\lambda^{(p)}$ is $a^q\lambda$ where $q$ is the smallest integer such that $R(a^q\lambda) < R[\lambda^{(p-1)}]$.

Note that $a^q\lambda$ increases with $q$. Thus, at step (c), there must be a value of $q$ satisfying the inequality. Indeed for any reasonable choice of $S_o$, $a^q\lambda$ will be of moderate size. At step (e), $a^q\lambda$ may eventually exceed the capacity of the machine. Detection of this situation is used to determine that the process has reduced $R$ to its local minimum value. (See Section 9, below.)

Examination of (7) shows that, for a given size of $\Delta x$, the size of $\lambda$ is governed by the size of the elements of $W_y$ and $W_x$. Typically, $W_x$ is diagonal, with greatest value along the diagonal equal to 1. $W_y$ has elements about $10^{-6}$. In this case a good first guess for $\lambda^{(1)}$ is $10^{-6}$. If $\lambda$ is started too low, the strategy immediately (i.e., within the first iteration) builds it up to a value consistent with the problem. If $\lambda$ is started too high, the first few iterations are wasted while the strategy allows $\lambda$ to decrease by means of step (a) which is applied only once per iteration. The parameters a and b both are currently chosen to be 10. Choices of a and b have not been studied from the viewpoint of optimization, but wide variations within the scope of (10) have negligible effect on the speed of convergence.

## 6. Constraints

The "end" conditions to be achieved fall into three classes:

Class 1. Parameters which have to achieve a given value

Class 2. Parameters which have to lie in a given interval

Class 3. Parameters which are to be minimized or maximized ("optimized")

Each of these classes is treated separately, as far as the corresponding values in the weight matrix $W_y$ and in the residual vector $\Delta y$ are concerned.

We introduce two new vectors $y_{min}$ and $y_{max}$, where each component of $y_{min}$ does not exceed the corresponding component of $y_{max}$:

$$y_{min} \leq y_{max}$$

At each stage of the iteration, the vector $y$ is compared, component by component, with both $y_{min}$ and $y_{max}$. If each component of $y$ satisfies the inequality

$$y_{min} \leq y \leq y_{max} \tag{11}$$

the procedure is terminated, and the current vector $x$ of independent variables is the solution to the problem (see Section 9).

In general, of course, (11) will not be satisfied. The vector $\bar{y}$ is described to the program solely in terms of $y_{min}$ and $y_{max}$:

$$\bar{y} = \frac{1}{2}(y_{min} + y_{max}) \tag{12}$$

The relationship between each element of $y_{min}$ and the corresponding element of $y_{max}$ now indicates to which class the element of $y$ belongs. If $y$ is of Class 1, the primary quantity is $\bar{y}$ and the corresponding elements of $y_{min}$ and $y_{max}$ are chosen close together:

$$y_{min} = \bar{y} - \frac{\delta}{2} \quad ; \quad y_{max} = \bar{y} + \frac{\delta}{2} \tag{13}$$

where $\delta$ $(>0)$ is a tolerance introduced for numerical purposes, since it is usually impossible to insist on absolute equality in digital computing. The value of $\delta$ for Class 1 parameters may be chosen as small as physical reality demands provided it is above the computational noise. The progress of the iteration or the solution obtained, in general, is not affected by this choice except that extremely small values may tend to increase the time for answers to be determined. In particular, the existence or nonexistence of a solution does not depend on the choice of $\delta$, so long as it is chosen consistent with the restrictions detailed above.

For elements of Class 2, the interval which must contain the y-value is the primary quantity and is used to describe the corresponding components of $y_{min}$ and $y_{max}$. That is, if the interval is $a \le y \le b$, we set

$$y_{min} = a \quad ; \quad y_{max} = b \tag{14}$$

for that element, and for purposes of residual computation $\bar{y}$ is determined from equation (12).

Now an important distinction between this class and Class 1 arises. In Class 2, any value within the interval is just as acceptable as any other. Thus, there is no merit in further reducing this component of $y - \bar{y}$ and therefore, at each stage of the iteration, if any component of $y$ belongs to Class 2, and if it is within its acceptable interval, the corresponding row of the matrix $P$, the corresponding row and column of the matrix $W_y$, and the corresponding contribution of this component of the $\Delta y$ vector to the residual $R$ are all deleted during the current iteration.

This action has two effects. First, application of the basic formula (8) no longer takes this particular parameter into consideration when it computes the correction $\Delta x$. Secondly, the current value of the penalty function $R$ contains no contribution from this parameter. If the corrected $x$ vector becomes such that a Class 2 parameter moves out of its interval of acceptability, $R$ will now include its effect. Thus, $R$ may be increased and, as seen in Section 5, the value of the inhibitor $\lambda$ will be increased. Thus, the correction is held back to a size small enough that the component of $\Delta y$ remains deleted from the value of $R$. In this case, we say that this dependent variable has approached a barrier.

Sometimes reintroduction of a component of $\Delta y$ into the computation of $R$ is accompanied by changes in other components of $\Delta y$, resulting in

a reduction of $R$ in spite of the reappearance of a previously deleted component of $\Delta y$. Naturally, in this case, the procedure cannot detect the presence of the barrier, and continues without increase in $\lambda$. Later iterations will, however, rectify this situation.

To mechanize this distinction between Class 1 and Class 2 parameters and to achieve the deletion of components, it is convenient to introduce a vector $C$ of switches $c$, each of whose $n$ elements has the value 1 or 0. If a dependent variable is of Class 1, the corresponding element of $C$ is 1; if it belongs to Class 2, the element is 0. (If the variable is of Class 3, the element of $C$ is also 0, but the distinction between Classes 2 and 3 is made on the basis of the values actually being computed.) Now when any dependent variable is within its interval, the weight matrix $W_y$ is replaced by a matrix $W_y'$ with the corresponding row and column multiplied by the component of $C$ corresponding to that variable. The matrix $W_y$ appears in all the computations involving the partial matrix $P$ or the residual vector $\Delta y$. Thus, when $W_y'$ is used in formulas (7) and (8) instead of $W_y$, the required deletions are automatic. Note again that this deletion is done separately for each iteration; that is, $W_y$ is always the matrix operated on by $C$, not the $W_y'$ matrix from the previous iteration.

A Class 3 parameter is designated by its relative weight, not by the method of computing residuals (see Section 7). However, for uniformity,

the interval of acceptability must still be established. This interval is chosen so that it cannot possibly contain the values of the parameters which are attainable when the Class 1 and 2 parameters are acceptable. If it is desired to minimize a variable, the $y_{max}$ element must be taken smaller than attainable; if the variable is to be maximized, the corresponding $y_{min}$ element must be taken larger than attainable. Usually, for Class 3 variables the corresponding values of $y_{max}$ and $y_{min}$ are equal, but this is not necessary.

## 7. Scale Matrices

The scale matrices $W_x$ and $W_y$, in addition to making dissimilar components compatible, are also used to represent the relative importance of the independent variables and of the dependent variables. In the most general situation $W_x$ and $W_y$ can be any symmetric, positive definite matrices of order m and n, respectively. Thus, for any vector $\Delta x$, the quantity $(\Delta x)' W_x \Delta x$ is defined and is a scalar with the property that

$$(\Delta x)' W_x \Delta x > 0 \tag{15}$$

unless $\Delta x$ is the zero vector. Corresponding statements can be made about $W_y$. Up to this time, experience with the iterator has been with diagonal $W_x$ and $W_y$, and such diagonal matrices have always proved adequate. Such a restriction is equivalent to assuming that the variables are uncorrelated.

A diagonal matrix is indeed positive definite if, and only if, all the elements on the diagonal are positive. Thus, $R = (\Delta y)' W_y \Delta y$ and $S = (\Delta x)' W_x \Delta x$ reduce to $\Sigma w_y (\Delta y)^2$ and $\Sigma w_x (\Delta x)^2$, that is, weighted sums of squares of residuals and corrections, respectively. We use $w_x$ and $w_y$ to denote entries on the diagonals of $W_x$ and $W_y$, respectively. Based on this meaning of R and S, it makes sense to speak of an individual term of either sum.

At each iteration, the size of the components of $\Delta x$ are adjusted so that the individual terms of $S$ are about equal. Similarly, near convergence, the size of the components of $\Delta y$ are such that the individual terms of $R$ are about equal. This equality, with regard to both the terms of $R$ and $S$, is in terms of program units. Thus, for example, an increment in time may be measured in hours and an increment in angle in radians. An error of $0.01$ hours in arrival time, for instance, is usually insignificant while an angle of $0.01$ radians may be quite significant (in calculating fuel used to execute a plane change, say). To equalize these effects, the user may choose $w_x$ values so that each change in $\Delta x$ has about the same relative significance. A smaller value of $w_x$ for any component of $\Delta x$ allows a larger amount of change in that component, and vice versa. Note, however, that the weight $w_x$ multiplies the square of $\Delta x$ so that multiplying $w_x$ by $100$, say, has the effect of reducing $\Delta x$ by only $10$. In addition, inspection of equation (8) shows that, if the inhibitor $\lambda$ is so small as to be negligible, differences in the sizes of $w_x$ have no effect.

In summary, then, the user controls the relative rates of convergence of the independent variables to their solutions by choosing the relative sizes of $w_x$. If there is more than one solution for any variable, the actual answer obtained may depend on these rates to a great extent. Otherwise, the choice of $w_x$ is not critical; only the speed of obtaining an answer is affected.

The values of $w_y$ govern the progress of each iteration, but their full effect is realized only near the converged answer. It is well to remember that equation (8) is used with a modified form of the matrix $W_y$. Thus, near convergence, several of the dependent variables y will be within regions of acceptability, and, consequently, deleted from equation (8) and from the calculation of the penalty function R. Those terms which remain in R are the ones which are about equal. Again, equality is in terms of program units. The user controls the relative size of the residuals of Class 1 and 3 variables as well as how close Class 2 variables are to their barriers (see Section 8) by his choice of $w_y$. A larger value for $w_y$ causes a smaller residual in a Class 1 variable or a closer hugging of the barrier in a Class 2 variable if, indeed, a barrier is reached. Smaller values for $w_y$ cause larger residuals in Class 1 variables and larger excursions from barriers in Class 2 variables. As a result, if he chooses, the user may get higher (or lower) optimum values for Class 3 variables by allowing more error in Class 1 and 2 variables. This effect may be achieved by increasing $w_y$ for Class 1 and 2 variables, or by decreasing $w_y$ for Class 3 variables.

If a problem requires the "optimization" of two or more variables, that is, requires more than one Class 3 variable, the user must assign, by proper choice of $w_y$, the relative importance of each one. In effect, a linear combination of these two variables will then be optimized.

The $w_y$ have another function in the early stages (selection mode) of the iteration. During selection, Class 3 variables must have weights $w_y$ so small that their contribution to R is negligible compared to the residuals of Class 1 and 2 variables. This is the really distinguishing feature of Class 3 variables. As long as any Class 2 variable is outside of its region of acceptability (up to a numerical tolerance), the corresponding terms in R must be larger than the terms for Class 3 variables. This will clarify the notion of inaccessible values for Class 3 variables in Section 6. Values are inaccessible if they are inaccessible while the Class 1 and 2 variables are within their regions of acceptability.

During selection, it may also happen that a Class 2 variable has a wide range of acceptability, but that it must be brought within that range before too many Class 1 variables get close to their respective desired values. This is due to the requirement that, at each iteration, the total penalty function R must not increase. Sometimes, if a Class 1 variable has a small residual term in R and if the Class 2 variable has a relatively large term in R, there is no way to bring the latter within its range of acceptability without causing the residual in the Class 1 variable to grow, in turn causing R to get larger. Consequently, a stand-off may result, with neither variable acceptable. To prevent this, a relatively large value of $w_y$ must be applied to the Class 2 variables of this type. Typical instances of such variables in the Apollo Mission Design are the various

inclinations, which have wide limits of acceptability, but which must be forced to within their interval of acceptability before any other residual makes a large contribution to equation (8).

As a general rule, one may say that the relative sizes of the several $w_y$ values represent the respective relative rates at which the dependent variables approach their final values. The ordering is, first the Class 2 variables mentioned above, then the remaining Class 2 variables, along with the Class 1 variables, and finally the Class 3 variable(s).

Mathematically speaking, only the relative sizes of $w_x$, $w_y$ and $\lambda$ are important. Their absolute sizes must be chosen so that equation (8) does not cause overflows. The particular procedure in use for solving equation (8) requires that the value of $\lambda$ satisfy

$$\lambda^m < 10^{38} \tag{16}$$

where $m$ is the number of independent variables. Care must be taken during both the selection and the optimization modes that $\lambda$ does not violate the above inequality. The usual approach is to take the various $w_x$ values equal to 1, except where special comparative weightings are needed. Thus, the size of the entries in the matrix $P$ governs the choice of the various $w_y$ to insure that the inequality (16) is not violated. It is wise to allow $w_y$ to be smaller than actually required by the inequality, so as not to terminate the procedure prematurely. It is important to note that, except for inequality

(16), the absolute size of $\lambda$, $w_x$, and $w_y$ is of no importance to the procedure; only the relative sizes are significant in determining the course and destination of the procedure.

## 8. Barriers

In Section 6 we introduced the notion of a barrier and how it arises. This section describes how barriers are detected and how the knowledge that a barrier exists is used.

A dependent variable $y^*$ is at a barrier if

(1) it belongs to Class 2;

(2) $y^*_{min} \leq y^* \leq y^*_{max}$ for that variable; and

(3) either

(a) $$\frac{y^*_{max} - y^*}{y^*_{max} - y^*_{min}} \leq \epsilon$$

or

(b) $$\frac{y^* - y^*_{min}}{y^*_{max} - y^*_{min}} \leq \epsilon$$

Condition (2) expresses the fact that the current value of $y^*$ is acceptable, and condition (3) states that the distance of $y^*$ from one or the other end-point of its interval of acceptability is within a fraction $\epsilon$ of it full range. The current value of $\epsilon$ is 0.002.

The dependent variables are scanned in order. The first combined occurrence of conditions (1), (2) and (3) marks the variable on which action is taken. Further variables are not examined.

The action taken depends on whether the iteration is in the selection mode, or in the optimizing mode. In the selection mode, there must be at least one dependent variable $y$ of Class 1 which is outside of its tolerance interval. If all variables in Class 1 are inside their respective tolerance intervals, the iterator is in the optimizing mode. The assumption is made here that the weights $w_y$ are so chosen that, at the time when all the Class 1 variables are near their desired values, the Class 2 variables are all within their regions of acceptability, or close enough to be brought in quickly. No violation of this assumption has ever been observed.

Now consider what happens if a dependent variable is at a barrier during the selection mode. This means that a Class 1 variable is not at its desired value, but that some Class 2 variable $y^*$ is in its acceptable range. To be as close as within $\epsilon$ of full range to an endpoint of the range means that except in very fortuitous combinations of circumstances, the corrections $\Delta x$ to the independent variable required to bring the Class 1 variable significantly closer to its desired value are such as to drive $y^*$ out of its acceptable range. Thus, in accordance with the procedure of Section 5, the value of the inhibitor has increased to a size big enough to keep the $y^*$ acceptable. If no further action is taken, the inhibitor $\lambda$ will continue to increase until the iteration is terminated because inequality (16) is violated before a solution is obtained. The assumption is made that another solution (combination of independent variables) exists, which is in

the vicinity of the other endpoint of the range of acceptability for $y^*$. Thus, having detected that $y^*$ is at a barrier in the selection mode, the following action is indicated:

(1) Five iterations are performed, without increasing the iteration counter.

(2) The current value of the inhibitor is reset to a very low value.

(3) During these iterations, the region of acceptability for $y^*$ is contracted to a single point whose value is the value at the endpoint of the acceptable interval opposite to the detected barrier.

(4) Before the first of these iterations, a new residual in $y^*$ is created. The previous value of the penalty function is corrected for this, and the resulting value is used in inequality (9).

(5) At the conclusion of the five iterations, the interval of acceptability is restored, and

(6) The current value of the penalty function is again corrected for this change, and used in inequality (9).

(7) The value of the inhibitor is again reset to a low value [not necessarily the same as in step (2)].

It is clear that, at the beginning of these iterations, a large value in the residual $\Delta y^*$ is built up, since its acceptable region is now different. Thus, all the independent variables will be pushed away from their previous values, so that there now exists a chance to iterate to the new solution. After five of these iterations, the variable $y^*$ is close to its new desired value and the remaining dependent variables are, generally, close to their required values. Steps (5), (6) and (7) merely restore the original requirements, and the iteration can now proceed.

In problems with a large number of variables of Class 2, several of these barrier conditions may arise; in each case the appropriate procedure is taken each time until a solution is found within all the intervals of acceptability.

In contrast to a rather uncomfortable feeling we have about reaching a barrier in the selection mode, we are happier when a barrier is reached during the optimizing mode. In this case we have determined that, out of all the possible values of $y^*$ within its interval of acceptability, the one which is associated with the optimum value of the Class 3 variables is, indeed, the one at the endpoint, or barrier, where we happen to be. If no action were taken, of course, as above, the inhibitor $\lambda$ would be increased and the procedure would be terminated. This would preclude any further optimization which could be achieved if other Class 2 variables were allowed to vary further, with this new constraint. Thus, we proceed to enforce this condition, that $y^*$ has for its interval of acceptability the single point whose value is the value of the barrier it had come to. Note that this is different from the procedure (3) in the selection mode, where we try to attain the opposite barrier; here we insist on staying at the current barrier.

Despite the fact that $y^*$ belongs to Class 2, its residual now can never be deleted from formulas (7) and (8), since it will never achieve exact equality with a given number in the computer. Thus, all further corrections $\Delta x$ are calculated so that $y^*$ continues to have almost the same value.

We may say that the independent variables move along the barrier in their domain. That is, the single point at the end of the interval of acceptability is transformed into a "surface" in terms of the independent variables $x$, on which all further sets of $x$ must lie.

As further barriers are reached, the same action is taken until the procedure terminates because now the combined number of Class 1 variables and Class 2 variables at barriers is equal to the number of independent variables -- that is, the system is just determined -- or until one of the other termination conditions occurs first.

The weighting factors on the Class 2 variable must thus be chosen with some care because they determine just how far away from the barrier the independent variables may wander in seeking to optimize the Class 3 variables. During this stage of optimization, no value of $y^*$ is acceptable, so that it may lie equally probably on either side of the newly-found required value. In other words, it may lie just inside or just outside the original interval of acceptability. The user must choose the relative weights for his Class 2 and 3 variables such that only tolerable excursions from acceptability are gotten. The smaller the weight for the Class 2 variable is, compared to that of the Class 3 variable, the further the variable can depart from its originally required interval. Of course, the user may also shrink his original limits of acceptability if he cares to, so that the final values are still actually acceptable, without the iteration knowing about it.

## 9. Termination Procedures

In this section, we will simply list the several modes by which the iteration procedure is terminated. Each of the following is a cause for termination.

a. An error has been detected in computing the dependent variables, using (1), during the first iteration. Here it is suspected that the input data are wrong; they must be corrected to proceed.

b. An error has prevented computation of the $P$ matrix at any stage. This depends on the particular method used to compute $P$, which is beyond the scope of the present report.

c. An error has prevented solution of equation (8) at any stage.

d. All of the dependent variables satisfy inequality (11);

$$y_{min} \leq y \leq y_{max} \tag{11}$$

e. The maximum number of iterations specified by the user has been exceeded.

f. The value of $\lambda$ has exceeded the capacity of the machine to compute $\det(P' W_y P + \lambda W_x)$, which implies a violation of inequality (16).

With reference to condition a, above, if the y-values cannot be computed after the first iteration, then too large a step was taken in the vector of independent variables. This is clear because, at previous iterations, the vector $y$ could indeed be computed. This is just another reason to multiply the inhibitor $\lambda$, by a, successively until the x-vector is close enough to its previous value that $y$ can still be computed. Compare Section 5.

## 10. An Example

The iterator described above can be applied to a tremendous variety of problems. In this section we describe just one of the many applications it has had. It is the problem of finding the initial conditions for a trajectory starting at some point on a circular parking orbit around the moon and ending in the vicinity of a point on the surface of the earth about four days later. To be more precise, the unknowns are

$x_1$ : increase in scalar velocity at transearth injection

$x_2$ : amount of plane change at transearth injection

$x_3$ : selenographic longitude of sub-spacecraft point at transearth injection

The Class 1 y variable is

$y_1$ : height of perigee if the orbit is assumed continued in a vacuum

The Class 2 y variables are

$y_2$ : distance -- along surface of the earth -- from designated site to osculating plane at reentry

$y_3$ : inclination of osculating plane at reentry

$y_4$ : range -- along surface of the earth -- from reentry to longitude of designated site

$y_5$ : time from transearth injection to landing

The Class 3 variable is

$y_6$ : percentage of weight consumed in performing the transearth injection

The $y_{min}$ and $y_{max}$ vectors are

| | min | max |
|---|---|---|
| $y_1$ | 43.7 miles | 48.3 miles |
| $y_2$ | - 690.5 miles | 690.5 miles |
| $y_3$ | $12^o$ | $40^o$ |
| $y_4$ | 1380.9 miles | 5753.9 miles |
| $y_5$ | 86 hours | 110 hours |
| $y_6$ | 0% | 0% |

The odd numbers used for the distances arise from the fact that the output data were in statute miles, while requirements were given in round numbers of nautical miles. The interval for $y_1$ represents a small tolerance of $\pm 2$ nm around the desired value of 40 nm. The intervals for $y_2$ and $y_4$ represent the geometry of the landing area, which must contain the landing point. The intervals for $y_3$ and $y_5$ are operational restrictions. There is a solution with a given inclination every 24 hours. This explains why the interval for $y_5$ is 24 hours long. The computable values for $y_6$ are all between 0 and 1, so that the interval for $y_6$ indicates that $y_6$ is to be minimized.

The weights $w_x$ are all 1. The $w_y$ are:

| | weight |
|---|---|
| $y_1$ | $4.8 \times 10^{-7}$ |
| $y_2$ | $4.8 \times 10^{-7}$ |
| $y_3$ | $3.1 \times 10^{-5}$ |
| $y_4$ | $4.8 \times 10^{-7}$ |
| $y_5$ | $4.8 \times 10^{-7}$ |
| $y_6$ | $7.3 \times 10^{-10}$ |

Table I presents the values of $\lambda$, $x$, and $y$ for each iteration. Note that the selection mode terminates at the seventh iteration. During iterations 7 through 12, the percentage of fuel used is reduced, due to the fact that the magnitude of the plane change $\Delta A$ is decreasing. The return inclination changes with $\Delta A$ and the optimum fuel consumption occurs for the extreme value of the inclination ($40^{\circ}$).

The optimum, however, occurs in the interior of the allowable reentry-range interval and the major optimizing changes of reentry range occur subsequent to the twelfth iteration.

Figure 1 displays the behavior of $\log_{10} \lambda$ for the first sixteen iterations. The shape is typical of the iteration process. At the beginning $\lambda$ behaves quite violently, reaching values which are quite high. As a solution (in the select mode) is approached, $\lambda$ decreases, only to increase again at the iteration where selection is computed, then staying relatively constant until each barrier is reached. At each barrier $\lambda$ increases moderately. Finally, although not shown, when optimization is computed (on the 27th iteration) $\lambda$ increases to a number bigger than the computer will accommodate.

Figure 2 displays the behavior of $X_2$ and $X_3$, while Figure 3 displays the behavior of $y_2$, $y_3$, and $y_6$. At iteration 13, all of the x and y variables have been slowed down. Thus, the scales in Figures 2 and 3 are expanded at the thirteenth iteration. Notice that at the fifteenth iteration there is another slow-down as the minimum value of $y_6$ is reached.

11. References

(1) D. D. Morrison, Methods for Nonlinear least squares problems and convergence proofs, Tracking Programs and Orbit Determination, Proc. Jet Propulsion Laboratory Seminar, (1960), pp. 1-9.

(2) D. W. Marquardt, An Algorithm for Least-Squares Estimation of Nonlinear Parameters, J. Soc. Indust. Appl. Math., Vol. II, No. 2, (June 1963), pp. 431-441.

TABLE I

| | λ | Δv fps | ΔA deg | Long. deg | Lat. S-mi | c.- r. S-mi | Incl. deg | r.- r. S-mi | Time hrs | LEM wt | % of wt. consumed |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $1x10^{-11}$ | 2500.0 | 0. | 220.0 | 6287.1 | 117.2 | 39.98 | 2903. | 100.09 | 33,174. | 22.000 |
| 2 | $8x10^{-13}$ | 2541.0 | - 3.29 | 227.7 | 1482.9 | 1227.7 | 38.70 | 9226. | 106.68 | 32,984. | 22.526 |
| 3 | $1.3x10^{-7}$ | 2575.0 | - 5.46 | 224.1 | 330.0 | 1093.6 | 34.03 | 5440. | 102.54 | 32,759. | 23.135 |
| 4 | $1.3x10^{-9}$ | 2591.0 | - 6.80 | 221.5 | 107.4 | - 282.3 | 30.02 | 2235. | 99.78 | 32,606. | 23.546 |
| 5 | $1.1x10^{-10}$ | 2592.0 | - 6.33 | 221.6 | 49.1 | - 205.2 | 31.35 | 2161. | 99.86 | 32,645. | 23.444 |
| 6 | $6.9x10^{-11}$ | 2592.0 | - 5.66 | 221.7 | 51.9 | - 80.3 | 33.25 | 2328. | 100.00 | 32,699. | 23.298 |
| 7 | $3.5x10^{-10}$ | 2592.0 | - 5.54 | 221.7 | 46.2 | - 55.9 | 33.57 | 2438. | 100.03 | 32,707. | 23.276 |
| 8 | $2.3x10^{-10}$ | 2592.0 | - 5.37 | 221.7 | 46.6 | - 19.0 | 34.07 | 2495. | 100.07 | 32,720. | 23.242 |
| 9 | $1.4x10^{-10}$ | 2592.0 | - 5.11 | 221.8 | 47.0 | 38.3 | 34.82 | 2584. | 100.13 | 32,738. | 23.192 |
| 10 | $9.2x10^{-11}$ | 2592.0 | - 4.72 | 221.8 | 48.1 | 126.9 | 35.90 | 2715. | 100.23 | 32,763. | 23.125 |
| 11 | $5.9x10^{-11}$ | 2592.0 | - 4.19 | 221.9 | 50.0 | 262.7 | 37.44 | 2914. | 100.37 | 32,795. | 23.039 |
| 12 | $3.8x10^{-11}$ | 2593.0 | - 3.49 | 222.1 | 52.9 | 467.2 | 39.50 | 3211. | 100.60 | 32,830. | 22.942 |
| 13 | $1.9x10^{-10}$ | 2593.2 | - 3.36 | 222.1 | 46.2 | 505.3 | 39.83 | 3357. | 100.63 | 32,835. | 22.930 |
| 14 | $9.9x10^{-10}$ | 2593.2 | - 3.34 | 222.2 | 46.1 | 512.4 | 39.90 | 3371. | 100.65 | 32,836. | 22.928 |
| 15 | $6.3x10^{-10}$ | 2593.2 | - 3.31 | 222.2 | 46.1 | 523.5 | 39.99 | 3389. | 100.66 | 32,837. | 22.924 |
| 16 | $2.6x10^{-8}$ | 2593.2 | - 3.31 | 222.2 | 46.1 | 523.7 | 39.99 | 3390. | 100.66 | 32,837. | 22.924 |
| 17 | $1.7x10^{-8}$ | 2593.2 | - 3.31 | 222.2 | 46.1 | 524.2 | 39.99 | 3390. | 100.66 | 32,838. | 22.924 |
| 18 | $8.5x10^{-8}$ | 2593.2 | - 3.31 | 222.16 | 46.1 | 524.2 | 40.00 | 3390. | 100.66 | 32,838. | 22.924 |
| 19 | $5.4x10^{-8}$ | 2593.2 | - 3.31 | 222.16 | 46.1 | 524.4 | 40.00 | 3391. | 100.66 | 32,838. | 22.924 |
| 20 | $3.5x10^{-8}$ | 2593.2 | - 3.31 | 222.16 | 46.1 | 524.6 | 40.00 | 3391. | 100.66 | 32,838. | 22.924 |
| 21 | $1.8x10^{-8}$ | 2593.2 | - 3.31 | 222.16 | 46.1 | 524.6 | 40.00 | 3391. | 100.66 | 32,838. | 22.924 |
| 22 | $1.1x10^{-7}$ | 2593.2 | - 3.31 | 222.16 | 46.1 | 524.7 | 40.00 | 3391. | 100.66 | 32,838. | 22.924 |
| 23 | $7.3x10^{-7}$ | 2593.2 | - 3.31 | 222.16 | 46.1 | 524.8 | 40.00 | 3391. | 100.66 | 32,838. | 22.924 |
| 24 | $3.0x10^{-6}$ | 2593.2 | - 3.31 | 222.16 | 46.1 | 524.8 | 40.00 | 3391. | 100.66 | 32,838. | 22.924 |
| 25 | $1.9x10^{-6}$ | 2593.2 | - 3.31 | 222.16 | 46.1 | 524.8 | 40.00 | 3391. | 100.66 | 32,838. | 22.924 |
| 26 | $1.2x10^{-6}$ | 2593.2 | - 3.31 | 222.16 | 46.1 | 524.8 | 40.00 | 3391. | 100.66 | 32,838. | 22.924 |

FIGURE 1

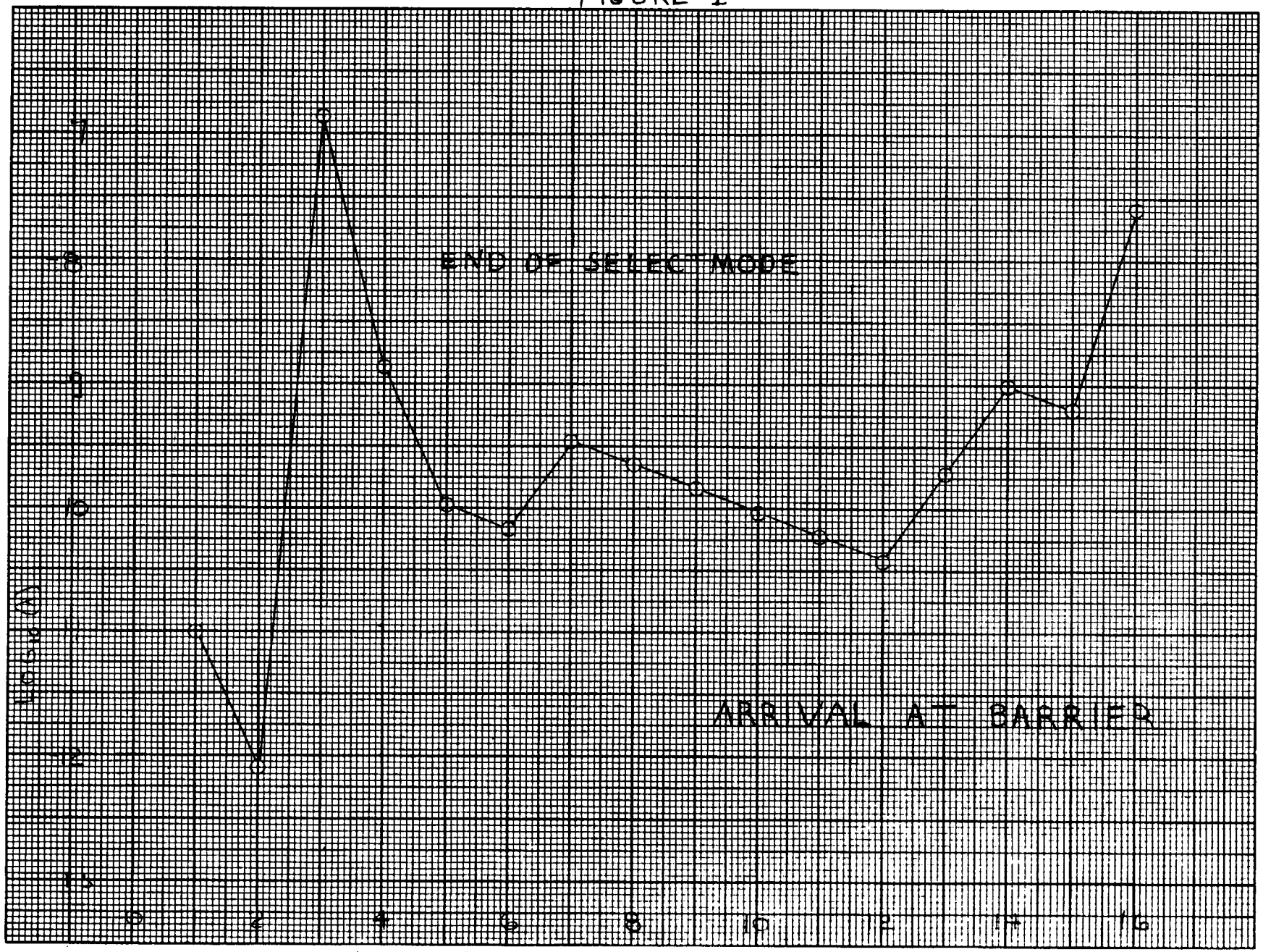


FIGURE 2

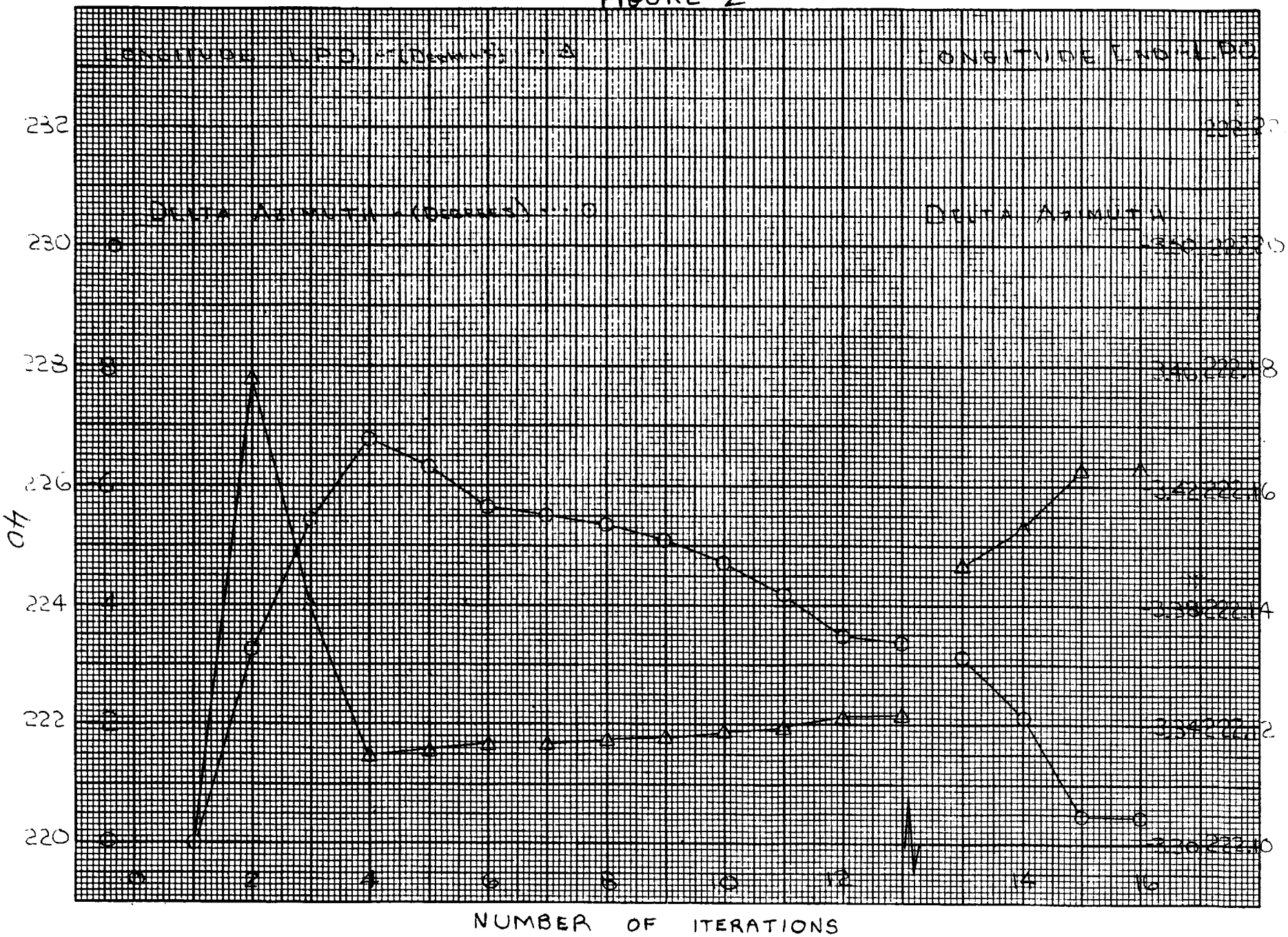
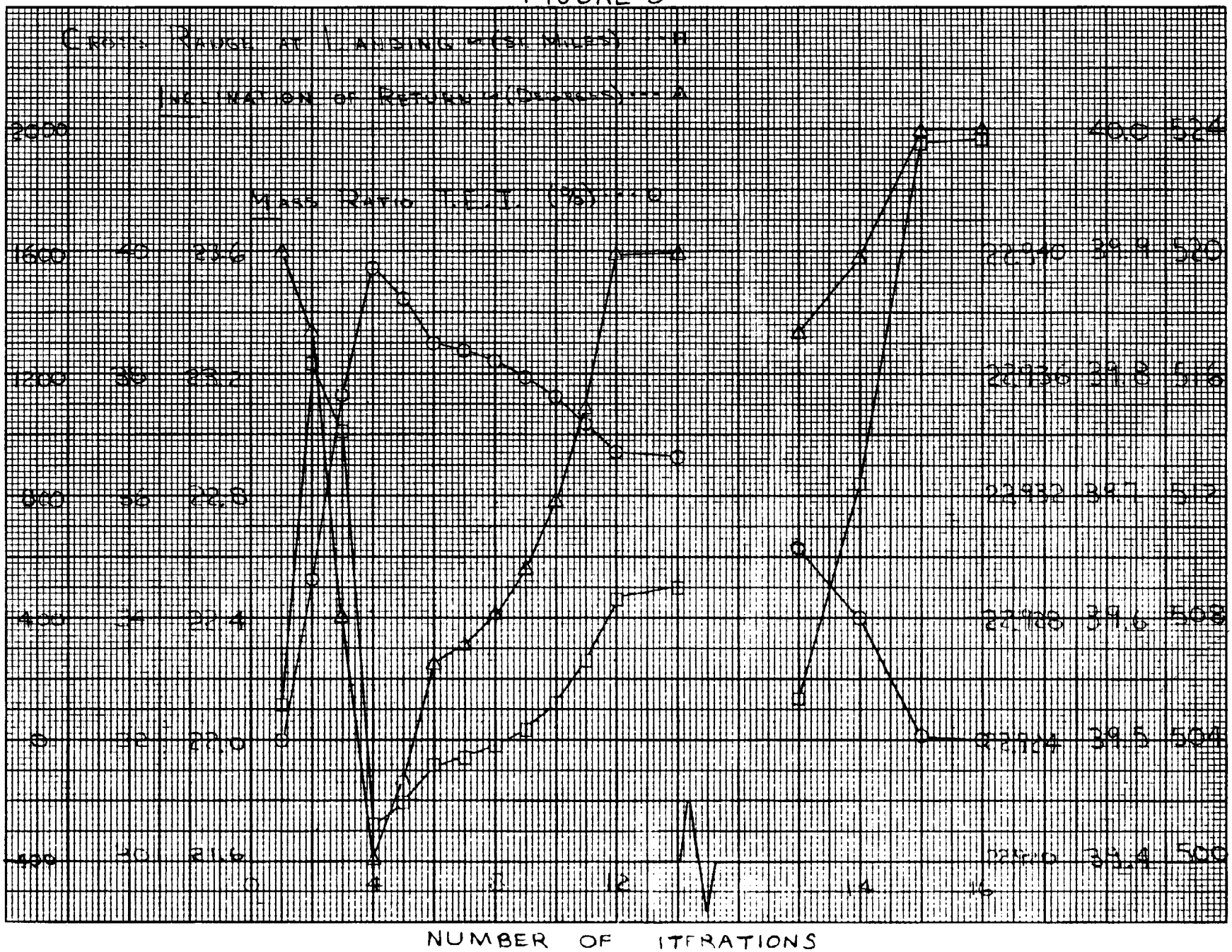
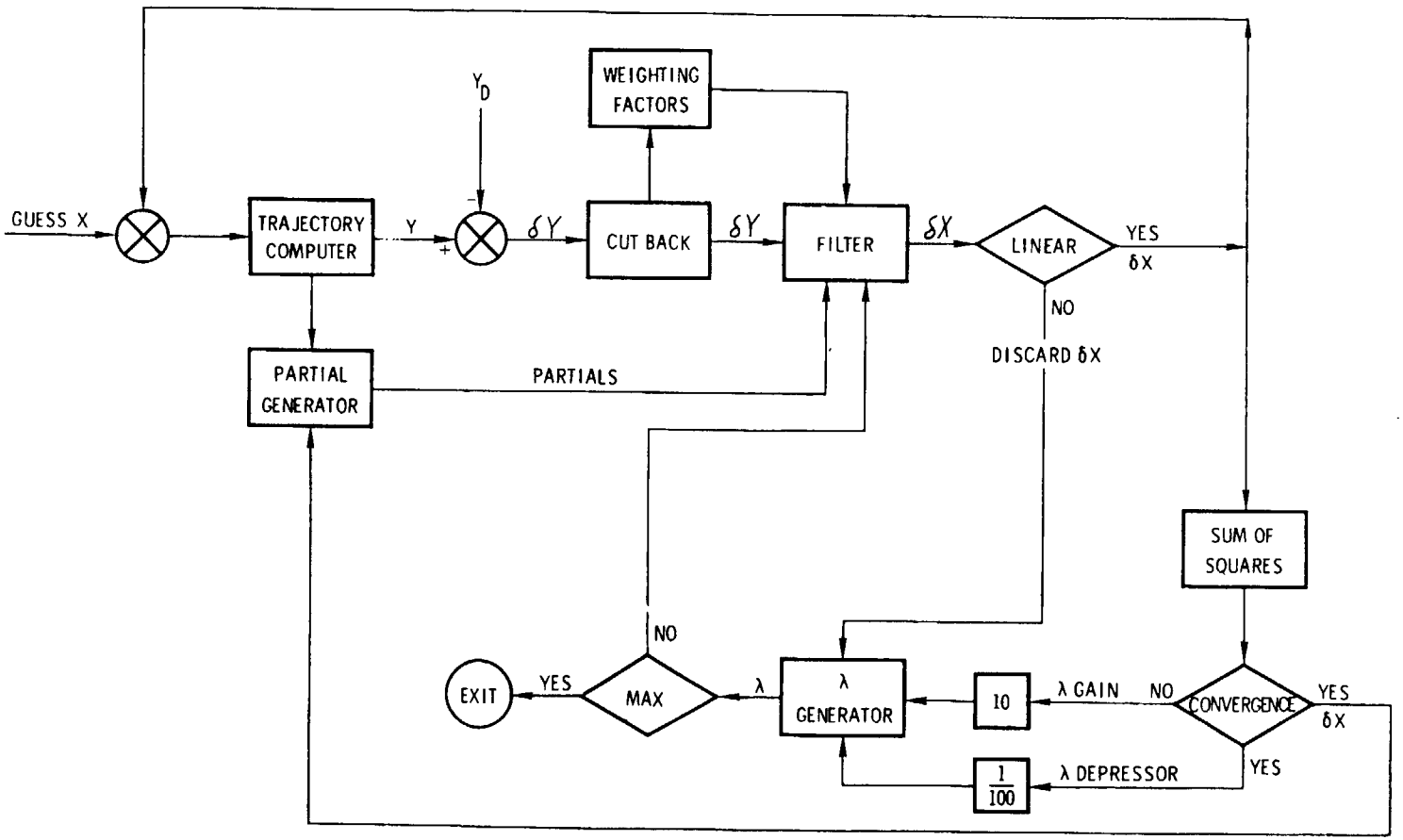

232
230
228
226
224
222
220
0
2
4
6
8
10
12
14
16
NUMBER OF ITERATIONS

FIGURE 3


CROSS RANGE AT LANDING = (ST. MILES) --- B
INCLINATION OF RETURN = (DEGREES) --- A
MASS RATIO T.E.T. (°R)
2000
1600
1200
800
400
0
-400
40
36
32
30
0
4
8
12
14
16
23.6
23.2
22.8
22.4
22.0
21.6
22340
22336
22330
22328
22314
22310
39.9
39.8
39.7
39.6
39.5
39.4
524
520
516
512
508
504
500
NUMBER OF ITERATIONS




ITERATOR LOGIC